the network police blotter

by Marcus J. Ranum

Marcus J. Ranum is CTO of Network Flight Recorder, Inc. He's the author of several security products and of a book on computer security (with Dan Geer and Avi Rubin) and is a part-time sysadmin.

<mjr@nfr.net>

Some Results

Mark Maris sent in a really nice haiku:

The extra bits fall
out onto the command line . . .
buffer overflow!

And another good one from Jason Testart:

On the bastion host,
a flurry of port scanning.
No process listens.

Hugh Kennedy gets honorable mention:

root password mumbled
aloud in a crowded lab.
how fast can i type?

I'm sending each of these worthy gentlemen a cool T-shirt that will make them the envy of their friends. I'm going to see if I can get them and the other contestants to let me publish their efforts on the Web. By the time you're reading this, I should have them online at *<http://pubweb.nfr.net/~mjr/usenix/haiku>*.

Building Burglar Alarms

My first "real" job was working for a burglar-alarm company when I was a kid in high school. Now, umpty-whatever years later, I'm basically doing the same thing for a living again. My boss at the time used to design great alarm systems; a lot of historic properties and municipal buildings in Maryland (including a nuclear-power plant) had alarm systems that had been designed specifically for those sites by my boss.

One of the distinguishing characteristics of his alarm systems was that he liked to put special traps within the perimeter of the site that was being secured – not just at the boundary of the network, um, uh, building. So the alarms had all of the usual glass-break detectors, window switches, and so on – with the addition of window switches hidden in gun cabinets, jewelry boxes, executive desks, key boxes, and so on. When we wired an alarm system with a bell, we'd put in a low-power circuit that would cause the alarm to go off (silently, of course!) if the bell circuit was cut. If the bell was in a box on the outside of the building, we'd include a contact switch in the bell box to set off the alarm if it was opened or pulled away from the wall.

Never mind infrared and microwave motion detectors (which are pretty obvious LCD-blinking white boxes stuck to a wall) – we used to put pressure-sensitive pads under carpeted floors in hallways. I guess I learned paranoia at an early age! Remember, one person's "paranoia" is another person's "engineering redundancy" – these traps worked extremely well. The special alarms would trigger a different code from the normal ones, so if the system called the police, the cops would know that they were dealing with a professional who had bypassed the first layer of security.

Obviously, you can see where I'm going with this: the same concepts apply to network and even application security.

If you've got a system that you want to keep people from breaking into, ask yourself a couple of leading questions:

- What am I really afraid could go wrong with my system?
- What would it look like if something were in the process of going wrong?
- What can I put in place to tell me when it is happening?

That's the simple recipe for a decent computer burglar-alarm system. Put alarms in place and I guarantee you eventually you'll come out looking like a hero – someone will break into your system using some ultra-sophisticated new trick, and you'll find out about it virtually instantly. While everyone else is running around in panic and confusion, you'll be the person on the spot armed with knowledge; it's a nice feeling!

The only drawback with burglar alarms is that we can't all use the same burglar alarms and have them work. To work correctly, the alarm has to be a surprise for the bad guy. Think of it as scattering land mines around your network. If we all put them in the same places, laid out in a nice, standard, neat grid, then the bad guys will instantly know where not to walk. So you need to be creative and think up your own burglar alarms. As tasteless as it seems, the land-mine analogy is actually very good because, like mines, burglar alarms are probabilistic devices. The more mines you put in a given area, the greater the likelihood someone will set one off if they enter that space. The more traps you put on your systems, the greater the likelihood someone will set one off if they enter your computer.

A Few Good Tricks

Most firewalls let you "overload" rules in front of one another. By placing a more specific rule "ahead" of general rules, you can cause the firewall to take different actions even against traffic within your own network. Suppose you have a Web server behind a screening router that is your boundary to the Internet. Consider placing screening rules in the router that "overload" the generic outgoing rules, so that if the Web server starts generating traffic to the outside that it normally wouldn't (say, IRC or ftp) it generates a log message. The rest of your network traffic gets different rules applied. Obviously, this only works if your Web server produces and consumes only a limited number of services. If it's running a zillion servers on various ports, then you may as well forget about security on it, anyhow.

Or suppose you have a Web server on a "DMZ" (semi-public) network outside of your firewall. Consider putting rules in your firewall that block and log traffic from the Web server trying to come in to your network. Give those rules higher precedence than the normal "block everything coming in" Internet rules. That way, it's a dead giveaway if someone breaks into your Web server and tries to probe for a way into your internal network. Consider installing in-kernel firewalls on perimeter machines such as Web servers or external DNS servers. Overload the firewall rules so that they generate alarms if the other systems on your DMZ network start probing one another. Have them generate alarms if they get traffic that should have been screened by your boundary router. Since none of these things should happen, the alarms should never fire. Right? You might be surprised some day.

Another common thing bad guys do when they get on your system is remove logs. If you're at all familiar with using make to build C programs, it's very easy to replace or add useful traps to your existing system software. For example, a fun thing to do would be to modify a long-running daemon so that it monitors the system log file and raises an alarm if the file ever vanishes or gets shorter during the day. Just hold an open descriptor to the file, and if the inode number returned by fstat is not the same as the inode number returned by stat, and/or the file size is different, then you know you have a problem. If you've still got the descriptor open and the bad guy forgot to truncate the

Another common thing bad guys do when they get on your system is remove logs.

"Are you trying to say that we should all quit trying to secure our networks and get jobs as plumbers or something?"

file and simply removed it, then you can lseek back to the beginning and write the file out somewhere safe. Essentially, you're trojan-horsing your own system – except that when *you* do it, it's legal and ethical.

If you're running a dedicated system that isn't for general-purpose use, you can replace system binaries with ones that raise an alarm if they are run. An extreme case would be replacing ls and more with programs that raise an alarm if they are ever run with privileges. Then train yourself to use echo with shell wildcard expansion, or some other means of seeing what's on your system. If you're running a multiuser or guest system, and you're not afraid of intruding, consider replacing networking programs such as telnet, ftp, and IRC with versions that log who ran them and to what they connected. You may wish to modify your kernel (if you're into that) to record stuff directly from the connect system call, as well as to record whenever the execute bit is set on a file. I recognize and respect the privacy issues such actions would raise; these ideas are not for everyone.

A few more ideas: many of the free UNIXes let you set an option to generate an error if something tries to execute code off the stack. There are also software solutions for detecting stack-smashing attacks from within the runtime environment. Consider turning such tools on, if you're managing a Web server, and you may be the first (or, actually, the second . . .) to know of the latest buffer-overrun attack. If you've got the time, it's not too hard to have a program monitor the process table and watch your Web server to see if anything weird happens to the process. Weird things that shouldn't happen: key processes vanishing, getting smaller, dramatically changing the amount of CPU they use, etc. You want to know these things, believe me! You probably don't want to have a process sitting there monitoring your process table – it'll stick out like a sore thumb. There are all kinds of fun places you can put it: as a subroutine of inetd, cron, or even sendmail, to name just a few.

Web-site defacement was all the rage in the press last year. It's still a big issue if it's your site that gets hit. Make sure you're the first person to know about it, if it happens. If you've got a friend on the outside with a high-speed connection, set up mutual watch scripts on your sites that monitor each other's root page (or pages) for changes. There are also Web-page-watching services that can be set up to email you if a page changes; have it watch your page and mail to an alphanumeric pager. That's probably not as fast, in terms of alert latency, as it should be, but it's cheap and easy.

The Plumber's Tale

I was giving a talk at a conference recently, and I'm afraid I was being a bit depressing and slightly cynical about the state of Internet security.

One of the people in the room asked a really thought-provoking general question, which was, "Are you trying to say that we should all quit trying to secure our networks and get jobs as plumbers or something?"

I have to admit that I was kind of speechless for a second, and mumbled something about never giving up, never surrendering, and so forth. It wasn't until later when I was sitting around with some friends that we collectively thought of a much better answer, namely that being a plumber isn't much of a job change.

As security or firewall administrators, we've got basically the same concerns: the size of the pipe, the contents of the pipe, making sure the correct traffic is in the correct pipes, and keeping the pipes from splitting and leaking all over the place. Of course, like

plumbers, when the pipes do leak, we're the ones responsible for cleaning up the mess, and we're the ones who come up smelling awful . . .

Next Up

Did a few of the dirty tricks I proposed above inspire you? Send me your favorite(s), and if yours is the best, I'll send you a totally cool Network Police windbreaker. Send them to me <*mjr@nfr.net*> with a subject line reading "dirty trick" and I'll sort through them for my column after the next one.